CS 470 Homework 2

Due by 11:00am on Thu, Feb 25, 2021

Submission instructions

Submit your assignment through the Canvas system. Upload a single ZIP archive file named **hw2_<your first name>_<your last name>.zip**, containing:

1) A folder named "src" that contains all the source code that must be runnable

2) A README.txt file inside the "src" folder explaining how to run your code and if any environments or packages are needed. Please also include Honor Code (see below for detail) in this file.

3) A .pdf report

4) A .txt file of the output results named "output.txt".

No email submissions are accepted. At the top of your program solution, include a section named "Collaboration statement" in which you acknowledge any collaboration, help, or resource you used or consulted to complete this assignment. You can use any programming language(s) to perform the computations and visualizations.

This homework requires to perform the frequent itemset mining tasks on the file data.csv described below. The file data.csv contains a dataset of the tweets in the year of 2014 related to the topic of flu shots. Each row corresponds to each tweet while each column denotes an attribute. This database contains 12 attributes, which are quite self-explained by their names. This homework requires us to focus on the attribute "text_keywords", but feel free to leverage other attributes' information for supporting your narrative in your report on frequent itemset mining results analysis.

According to Emory Honor Code:/* THIS CODE IS MY OWN WORK, IT WAS WRITTEN WITHOUT CONSULTING CODE WRITTEN BY OTHER STUDENTS. Your Name Here */

Code. To accomplish this task, you will use the attribute "text_keywords", which is a set of keywords separated by ";" for each row. Other attributes or columns are not needed. Here each keyword is considered as an item; for each row, the set of keywords can be considered as a "transaction". Using the Apriori method, our goal is to find frequent itemsets, namely keyword co-occurrence patterns that are very important patterns in text mining. Implement the Apriori algorithm for frequent itemsets mining, either in Java, Python, R, or MATLAB. The algorithm's pseudocode and related procedures are in the course slides and textbook. You are encouraged to use existing (e.g., those introduced in the class) or your own optimization techniques to speed up the algorithm. Explain and discuss the techniques you used, and provide the appropriate references and explanations on the technique selection.

Your program should be executable with 3 parameters: the name of the input dataset file (namely our file "data.csv"), the threshold of minimum support count, and the name of the output file as "output.txt" (in that order). The minimum support count is an integer. An itemset is frequent if its support count is greater than or equal to this threshold. You can tune the threshold of minimum support count for your algorithm on our data and find an appropriate value to generate your output file. Remember there is a

trade-off between efficiency and number of frequent itemsets generated from your algorithm. If your threshold is too small then your algorithm may need to run for a long time, while if it is too large, you may get too few frequent itemsets with little interestingness.

Output file. Your "output.txt" file should contain all the frequent itemsets together with the minimum support count you prefer. The output file should have the following format: each line contains a single frequent itemset as a list of items separated by a single space. Its support count is included between a pair of parentheses at the end of the line. For example: "home flu shot (530)" (without quote symbols in your file) represents an itemset containing items home, flu, shot with a support count of 530. In addition, please rank the lines of itemsets in descending order of their support counts. An example of output is provided in the file example-output.txt.

Report file. Write a report in a .pdf file presenting your results in your .txt file. Report the threshold of frequent count you chose to generate your output file and explain why. Explain and discuss, if any, the algorithmic optimizations you have used in your implementation. Discuss the experiences and lessons you have learned from this assignment. Report the minimum support value you use and analyze the results and discuss about what knowledge you can learn from the patterns your implemented method has found.

Hint: Please start early and be warned that an implementation without careful planning or efficient data structures could run for days! Excessively slow implementations will receive zero points (see grading criteria section) for the corresponding task. There are a few online repositories for frequent pattern mining implementations, such as http://fimi.uantwerpen.be/src/. You can study them but you are asked not to copy their implementations for this assignment. Remember that the Honor Code applies, and an automatic plagiarism checker will be used on submissions.

Grading criteria

- 50 points for code implementation: -10 for minor mistakes; -20 if there is some mistake but the program still works in most cases; -30 for more serious mistakes but the program still works in several cases. Zero points if the program does not compile following your readme file's instruction, or if the program compiles but gives mostly the wrong results or crashes. Zero points if the program takes longer than 10 minutes on our dataset in data.csv with minimum support count 500, on a CPU equal or faster than 2.3 GHz 8-Core Intel Core i9.
- 10 points for your results file "output.txt". -3 point if the file content format is wrong or not in descending order. -2 if using wrong file name. -5 if there is some mistake in the output, e.g., if there is ANY frequent itemsets that are missed.
- 40 points for a complete, clear, and well-organized report on analyzing your code implementation and results file "output.txt".
- *Note: 10 points will be deducted if your submission misses Honor Code or Collaboration statement each. If you miss both, -20.